Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anja Bisako

Uvedba potisne tehnologije v sistemu za naročanje spletnih storitev v oblaku

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: prof. dr. Borut Robič

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite različne programske rešitve za realizacijo potisne tehnologije. Na podlagi tega izberite najprimernejšo programsko opremo za razvoj prototipa. Slednji naj služi kot osnova za nadgradnjo obstoječega sistema za naročanje in upravljanje spletnih storitev v oblaku.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Anja Bisako, z vpisno številko **63070043**, sem avtorica diplomskega dela z naslovom:

Uvedba potisne tehnologije v sistemu za naročanje spletnih storitev v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom prof. dr. Boruta Robiča.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 24. septembra 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IRC	internet relay chat	internetni klepet
XMPP	extensible messaging and presence protocol	razširljiv protokol sporočanja in prisotnosti
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
CGI	common gateway interface	skupni prehodni vmesnik
SSE	server-sent events	dogodki, poslani s strežnika
HTML	hypertext markup language	jezik za označevanje nadbesedila
RSS	rich site summary	zgoščeni povzetek strani
AJAX	asynchronous JavaScript	asinhroni JavaScript
TCP	transmission control protocol	protokol za nadzor prenosa
MVC	model-view-controller	model-pogled-krmilnik
BSD	Berkeley software distribution	distribucija programske opreme Berkeley
DRY	don't repeat yourself	ne ponavljaj se
GPL	general public licence	splošno javno dovoljenje
ORM	object relational mapper	objektno relacijski preslikovalnik
IP	internet protocol	internetni protokol
API	application programming interface	vmesnik uporabniškega programa
JVM	Java virtual machine	Java virtualni stroj
SQL	structured query language	strukturiran povpraševalni jezik

Povzetek

Potisna tehnologija je mehanizem za pošiljanje podatkov s spletnega strežnika v spletni brskalnik brez zahteve po prenosu s strani odjemalca. Zaradi naraščajoče potrebe po komunikaciji v realnem času in čim boljši uporabniški izkušnji vedno več produkcijskih sistemov vidi prednosti v takšnem načinu podajanja informacij. V okviru diplomske naloge je bil opravljen pregled programskih rešitev za realizacijo potisne tehnologije. Na podlagi zaključkov je bila izbrana najustreznejša programska oprema in izdelan prototip, ki bo služil kot nadgradnja obstoječega sistema za naročanje in upravljanje spletnih storitev računalništva v oblaku.

Ključne besede: Potisna tehnologija, Django, Twisted, strežnik, odjemalec, podatkovna baza, sistem za naročanje.

Abstract

Push technology is a mechanism for sending data from a web server to a web client, where the client makes no explicit requests for the transfer. Due to the increasing need for real-time communication and optimization of the user experience there is a rising number of production systems that seize the advantage of this kind of information transfer. In the thesis we reviewed the existing software solutions for the realization of push technology. Based on the conclusions, we have selected the most appropriate software and developed a prototype that will serve as an upgrade to the existing system for ordering cloud services.

Keywords: Push technology, Django, Twisted, server, client, database, ordering system.

Poglavje 1

Uvod

V diplomski nalogi je obravnavano področje spletnih storitev s potisno tehnologijo. Problematika je postavljena v okvir izboljševanja obstoječega sistema za naročanje in upravljanje storitev računalništva v oblaku. V okviru rešitve obstaja več delovnih tokov, kjer med oddajo naročila in izvedbo naročila lahko preteče dlje časa. Posodabljanje statusa naročil na spletni strani trenutno poteka na zahtevo sproženo prek odjemalca, zaradi česar uporabniška izkušnja ni najboljša. Obstoječo rešitev, katere prednost je bila preprosta implementacija tako na strani odjemalca kot na strani strežnika, želimo nadgraditi, da bo omogočala bolj sprotno in uporabniku prijazno posodabljanje statusa naročil. Slabost rešitve, ki samodejno pošilja poizvedbe na strežnik, je, da je njena natančnost odvisna od pogostosti poizvedovanja. Če želimo doseči vtis, da posodabljanje statusov poteka sproti, posledično prožimo večje število poizvedb, ki povečujejo obremenitev spletnega strežnika, hkrati pa v veliki večini primerov niso potrebne, ker se od zadnje poizvedbe stanje ni spremenilo.

Izboljšana rešitev je osnovana na temelju potisne tehnologije, ki omogoča sprotno posodabljanje stanja na strani odjemalca, ne da bi se pri tem ustvarjalo nepotrebno breme na strani spletnega strežnika. V okviru diplomske naloge je opravljen osnovni pregled programskih rešitev za realizacijo potisne tehnologije, na osnovi katerega je izbrana programska oprema za imple-

mentacijo nadgradnje. Pri izbiri osnovne programske opreme je pomemben kriterij združljivost s trenutno programsko opremo - pri izbiri imajo prednost rešitve, ki so preverjeno združljive s trenutno rešitvijo.

Praktični rezultat diplomske naloge je prototip, ki omogoča demonstracijo izboljšane uporabniške izkušnje s sprotnim posodabljanem informacij o stanju posameznih naročil. Pri oblikovanju rešitve je posebna pozornost osredotočena na skalabilnost rešitve in možnost integracije zalednega dela rešitve z različnimi izvori informacij. Zagotavljeno mora biti normalno delovanje in hitra distribucija podatkov tudi v okoliščinah, ko število aktivnih uporabnikov dosega okvire več sto tisoč uporabnikov.

Izbrano programsko rešitev želimo kasneje vgraditi oziroma povezati z jedrom obstoječe rešitve, tako da so vse spremembe stanja naročil objavljene na novem komunikacijskem vodilu. Spletne strani, ki vključujejo informacijo o stanju naročil, so povezane s komunikacijskim vodilom in dopolnjene z ustrezno programsko logiko, ki skrbi za posodabljanje podatkov na osnovi sporočil, prejetih prek komunikacijskega vodila. Rešitev želimo kasneje v praksi uporabiti še na drugih delih sistema, zato mora arhitektura rešitve omogočati preprosto povezovanje različnih sistemov na vodilo za distribucijo podatkov. Pomembna je tudi preprosta integracija komunikacijskega vodila v spletne strani in zanesljivo delovanje v različnih spletnih brskalnikih.

Poglavje 2

Potisna tehnologija

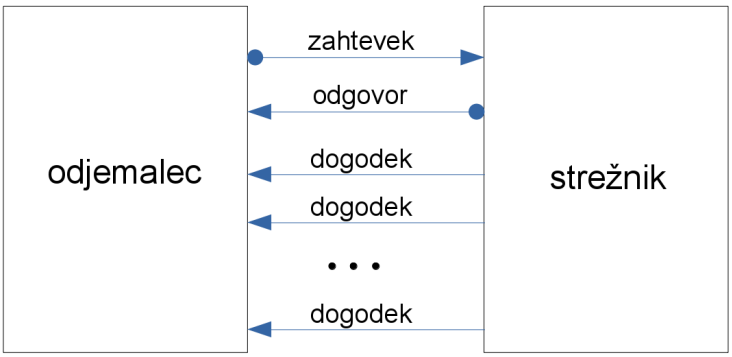
Potisna tehnologija *(angl. server push)* je oblika komunikacije, ki uporabniku omogoča avtomatsko dostopanje do vsebin z interneta v predpisanih intervalih ali na podlagi uporabniško določenih kriterijev. V nasprotju s tradicionalno vlečno tehnologijo, kjer je zahteva po prenosu informacije sprožena s strani odjemalca, pri potisni tehnologiji akcijo sproži strežnik, ki pove, kje se določena vsebina nahaja in jo potisne na odjemalca. Skica delovanja potisne tehnologije je prikazana na Sliki 2.1.

Potisne storitve pogosto temeljijo na vnaprej določenih preferencah. Temu pravimo model za objavljanje in naročanje vsebin, kjer se odjemalec naroči na želene informacijske kanale, ki jih ponuja strežnik. Takoj, ko je na voljo nova vsebina na katerem od teh kanalov, strežnik potisne to informacijo na odjemalca. Primeri potisnih storitev so instantno sporočanje, internetni klepet ali video konferenca, ki temeljijo na protokolih kot so IRC in XMPP in omogočajo izmenjavo sporočanja v realnem času. Poznamo več različnih tehnik za realizacijo potisne tehnologije.

Long polling ni prava potisna tehnologija, ampak je variacija tradicionalne vlečne tehnologije, ki dovoljuje posnemanje potisnega mehanizma v razmerah, kjer potisna tehnologija ni mogoča, kot npr. spletne strani z varnostnimi politikami, ki zahtevajo zavrnitev dohodnih zahtevkov HTTP/S. Pri tej tehniki odjemalec zahteva podatke od strežnika na enak način kot

pri vlečni tehnologiji, s tem da so zahtevki HTTP/S na veliko počasnejši frekvenci. Če ob prihodu zahtevka strežnik nima na voljo podatkov za odjemalca, pusti zahtevek odprt in čaka, da so odzivni podatki na voljo. Ko se to zgodi, strežnik takoj pošlje odgovor HTTP/S odjemalcu in zaključi odprto zahtevo. Na ta način je odpravljena latenca odgovora, značilna za vlečno tehnologijo (čas, ki poteče med trenutkom, ko so podatki prvič na voljo in naslednjo zahtevo odjemalca).

Pushlet je tehnika, pri kateri strežnik izkoristi obstojne povezave HTTP in pusti odziv večno odprt, kar prelisiči brskalnik, da ostane v stanju nalaganja spletne strani. Strežnik nato pošilja majhne dele kode JavaScript *(angl. snippets)*, da posodobi vsebino strani in s tem doseže zmogljivost potisne tehnologije. Z uporabo te tehnike odjemalec ne potrebuje programčkov Java *(angl. applets)* ali drugih vtičnikov, da ohrani odprto povezavo s strežnikom - odjemalec je samodejno obveščen o novih dogodkih. Slabost te tehnike je pomanjkanje nadzora strežnika nad časovno omejitvijo brskalnika. Če se časovna omejitev izteče, je vedno potrebna osvežitev strani.



Slika 2.1: Skica delovanja potisne tehnologije.

Potisna tehnologija HTTP (tudi pretakanje HTTP) je mehanizem za pošiljanje asinhronih podatkov s spletnega strežnika v spletni brskalnik. Potisna tehnologija HTTP je lahko realizirana na več načinov. Po tem ko strežnik postreže odgovor odjemalcu, običajno ne prekine povezave, ampak

jo pusti odprto, da so lahko v primeru kakršnegakoli dogodka podatki nemudoma poslani in ni potrebno čakati v vrsti na naslednji prejet zahtevek odjemalca. Večina spletnih strežnikov ponuja to funkcionalnost preko skupnega prehodnega vmesnika *(angl. CGI)*, ki opredeljuje skupek pravil, ki določajo, kako nek informacijski strežnik komunicira z lokalno programsko opremo in obratno. Z razvojem HTML5 se je uveljavila tehnologija SSE, ki je del standarda HTML5. Pri SSE brskalnik prejme samodejne posodobitve od strežnika preko povezave HTTP. Druga s HTML5 povezana tehnologija za realizacijo potisne tehnologije HTTP pa je API spletne vtičnice *(angl. WebSocket)*, ki omogoča popolno obojestransko komunikacijo TCP *(angl. full-duplex)* med spletnim strežnikom in odjemalcem. [1]

2.1 Razvoj potisne tehnologije

Potisna tehnologija se je začela razvijati leta 1996. Prvi val potisne tehnologije se je začel v obliki spletnega razširjanja programskih vsebin, znanem pod imenom webcasting. Pri spletnem razširjanju programskih vsebin je bil kanal povezan z neko kategorijo informacij, uporabnik pa je po registraciji v enem ali več kanalih začel avtomatsko prejemati ustrezne informacije, ki so bile prikazane s pomočjo namenske programske opreme odjemalca. Podjetje PointCast je prvo razvilo produkt, ki je postal popularen zaradi uporabe potisne tehnologije v obliki spletnega razširjanja programskih vsebin. PointCast Network je dostavljal novice in borzne podatke. Kljub temu, da so tehnologijo kmalu začela uporabljati tudi podjetja kot je Microsoft in Netscape, se zaradi prevelike porabe pasovne širine ta vrsta potisne tehnologije ni obdržala. Brskalniki so jo nadomestili z vlečno tehnologijo RSS.

V letu 2000 se je zaradi uspeha na področju spletnih sistemov varnega trgovanja ponovno pojavila potreba po potisni tehnologiji v realnem času. Potisnjeni podatki so se posodabljali na nivoju posameznega polja in v realnem času - manjša kot je bila latenca, boljša je bila kakovost platforme. Začetniki na tem področju so bili Ligthstreamer, Pushlets in Knowhow. Ta

oblika potisne tehnologije se je nenehno razvijala z dodajanjem novih funkcij že uveljavljenim rešitvam, z izboljšanjem zanesljivosti in učinkovitosti ter s predlogi za nove standarde. Zaradi dinamične narave tega nabora tehnologij se je potisna tehnologija v tej obliki obdržala in se razvija še danes. Vedno več produkcijskih sistemov vidi prednosti v dostavljanju informacij v realnem času. Potisna tehnologija se uporablja tudi pri igralništvu, spletnih stavah, v vojski in vesoljskih programih. Za potisno tehnologijo so se zaradi različnih tehnoloških potreb uveljavili številni izrazi, kot so Comet, AJAX push, reverse AJAX. Potisni kanal, ki vodi od strežnika k odjemalcu, je lahko implementiran z uporabo različnih tehnik, kot so polling, long-polling, frame streaming, Flash streaming, XHR streaming. Ustvarjeni so bili tudi številni protokoli, ki uporabljajo eno ali več omenjenih tehnik.

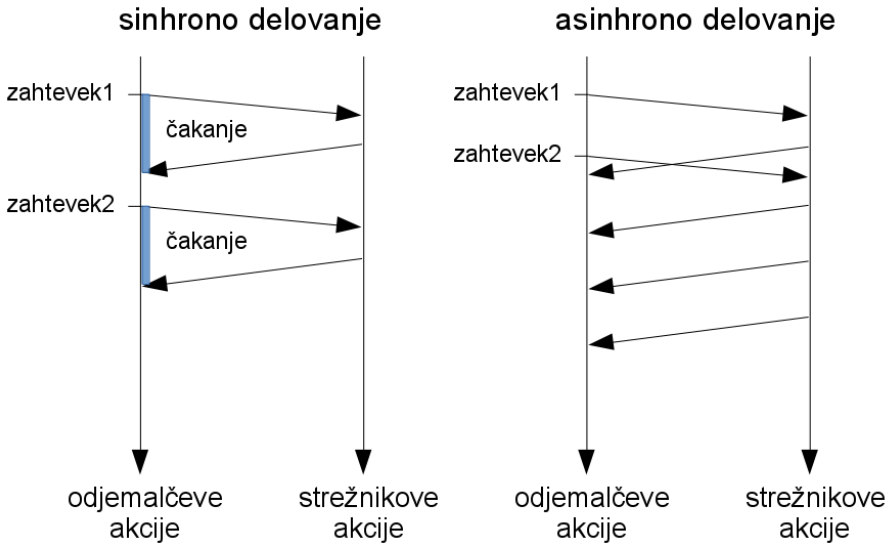
Vzporedno se je leta 2007 pojavilo zanimanje za dvosmerno komunikacijo, kar je vodilo v razvoj nove veje potisne tehnologije. Dvosmerna komunikacija omogoča potiskanje informacij v realnem času s strežnika na odjemalca in tudi z odjemalca na strežnik. Povratna informacija z odjemalca na strežnik je nov koncept in je pomemben zaradi zmožnosti pošiljanja sporočil z nizko latenco in visoko frekvenco z uporabo protokola WebSocket, ki omogoča popolno obojestransko komunikacijo preko ene povezave TCP. [2]

2.2 Pregled programskih rešitev

Delovanje običajne vlečne tehnologije je sinhrono. Odjemalec pošlje zahtevek strežniku, strežnik pa mu pošlje odgovor. Odjemalec mora počakati na odgovor strežnika, preden mu lahko pošlje nov zahtevek. Pri potisni tehnologiji pa želimo pošiljati podatke asinhrono, da lahko strežnik nenehno pretaka podatke odjemalcu. Razlika med sinhronim in asinhronim delovanjem je prikazana na Sliki 2.2.

Navadni strežniki, kot je npr. Apache, so zgrajeni za delo z veliko kratkoživečimi povezavami HTTP. Tak strežnik mora vzdrževati povezavo za vsakega odjemalca. Pri potisni tehnologiji pa imamo veliko dolgo-živečih po-

vezav in zato potrebujemo logiko za asinhrono pošiljanje sporočil. Tukaj se navadni strežniki srečajo s procesnimi omejitvami pri servisiranju. Potreben je zelo prilagodljiv in visoko razširljiv strežnik z dogodkovno arhitekturo, ki odpravi potrebo po vzdrževanju povezave za vsakega odjemalca z ustvarjanjem nove niti *(angl. thread)* in lahko vzdržuje več tisoč hkratnih uporabnikov. Potrebujemo torej strežnik, ki omogoča asinhrono delovanje in podpira večje število sočasnih povezav odjemalcev. [3]



Slika 2.2: Skica sinhronega in asinhronega delovanja.

Poleg primernega strežnika potrebujemo tudi komponento na strani odjemalca, ki poskrbi za povezavo s strežnikom. Običajno sta za ta namen uporabljena programski jezik JavaScript in knjižnica jQuery. V primeru, da se odločimo za implementacijo potisne tehnologije z uporabo API spletne vtičnice, ki nam omogoča popolno obojestransko komunikacijo med strežnikom in odjemalcem, moramo biti pozorni na kompatibilnost z brskalniki. Ker vsi brskalniki še nimajo implementirane podpore za spletno vtičnico, je najbolje, da za realizacijo takšne povezave uporabimo ustrezno knjižnico, ki poskrbi za ustrezno obnašanje v primeru slabe podpore brskalnika.

2.2.1 Rešitve na strani strežnika

Poznamo kar nekaj primernih spletnih ogrodij za realizacijo asinhronega delovanja spletne aplikacije v realnem času na strani strežnika. Ogrodja se razlikujejo v programskem jeziku, v katerem so napisana in v funkcionalnostih ter razširljivosti, ki jo ponujajo. Omenili bomo samo nekatera najbolj znana ogrodja, ki podpirajo več platfom, so razširljiva in večkrat tudi osnova za razvoj novih ogrodij. Pri odločitvi, katero ogrodje uporabiti za določeno aplikacijo, igrajo veliko vlogo funkcionalnosti, ki jih želimo implementirati, programski jezik, v katerem je aplikacija napisana in želena zmogljivost.

Node.js

Node.js je izvajalno okolje za strežniške in omrežne aplikacije. Aplikacije so napisane v programskem jeziku JavaScript in se lahko izvajajo znotraj izvajalnega okolja Node.js na operacijskih sistemih Linux, Microsoft Windows in OS X. Node.js uporablja vhodno-izhodni asinhroni dogodkovni model brez blokiranja, ki je primeren za razvoj aplikacij v realnem času. Ker ima vgrajene asinhrone vhodno-izhodne knjižnice za datoteke, vtičnice in komunikacijo HTTP, se aplikacije lahko obnašajo kot samostojen spletni strežnik. Node.js je priznan kot visoko zmogljiva strežniška platforma in ga uporabljajo Groupon, Microsoft, LinkedIn, PayPal in mnogi drugi. [4]

Twisted

Twisted je dogodkovno vodeno omrežno programsko ogrodje, napisano v Pythonu in licencirano pod odprtokodno licenco MIT. Bazira na dogodkovni programski paradigmi, kar pomeni, da ogrodje kliče kratke povratne klice, ki jih pišejo uporabniki. Zasnovan je za popolno ločitev med logičnimi protokoli in fizičnimi prenosnimi plastmi. Je izredno hiter, saj temelji na preloženčih *(angl. deffereds)* - mehanizem, preko katerega dosežemo asinhrono arhitekturo. Poleg tega imamo stalen nadzor nad zmogljivostjo izvajanja, vgrajeno podporo za samodejno razširjanje in lokalno multiprocesiranje ter podporo

za številne protokole. Twisted ogrodje uporabljajo med drugim tudi Sage, Tor2web in Apple koledar. [5]

EventMachine

EventMachine je sistem za pisanje visoko razširljivih aplikacij v programskem jeziku Ruby. Zagotavlja dogodkovno voden vhodno-izhodni mehanizem z uporabo vzorca za obdelavo zahtev za storitve, ki so dostavljene sočasno z enega ali več vhodov. Je najbolj priljubljena knjižnica za sočasno računanje v programskem jeziku Ruby. EventMachine je osnova za spletni ogrodji Goliath in Cramp. [6]

Tornado

Tornado je visoko zmogljivo odprtokodno razširljivo spletno ogrodje in asinhrona omrežna knjižnica. Napisan je v programskem jeziku Python in je bil prvotno razvit za FriendFeed. Uporablja neblokirajoč omrežni vhod-izhod, zato lahko vzdržuje več deset tisoč odprtih povezav, kar je primerno za delo z dolgo-živečimi povezavami. [7]

Vert.x

Vert.x je dogodkovno aplikacijsko ogrodje, napisano v več programskih jezikih (Java, JavaScript, Groovy, Python, Ruby, ...), kar pomeni, da so lahko komponente aplikacije napisane v kateremkoli od naštetih jezikov. Vert.x teče na JVM in je zgrajen na preprostem modelu vzporednosti, kjer je vsa koda enonitna. Zaradi preprostega, asinhronega modela programiranja je primeren za pisanje neblokirajočih razširljivih aplikacij. [8]

Libevent

Libevent je programska knjižnica, ki omogoča asinhrono obveščanje o dogodkih. Napisan je v programskem jeziku C in licenciran pod licenco BSD.

Omogoča razvoj prenosnih aplikacij in zagotavlja najbolj razširljiv mehanizem za obveščanje o dogodkih za operacijski sistem. Najbolj znane aplikacije, ki uporabljajo libevent, so Google Chrome, Transmission, ntpd in Tor. [9]

2.2.2 Rešitve na strani odjemalca

Na strani odjemalca potrebujemo kodo, ki bo poskrbela za prejemanje sporočil od spletnega strežnika. Običajno so to koda JavaScript in dodatne knjižnice, ki omogočajo lažjo implementacijo kode, ki je kompatibilna z različnimi brskalniki.

JavaScript

JavaScript je dinamičen skriptni programski jezik, ki se najpogosteje uporablja za implementacijo ukaznih datotek *(angl. script)* na strani odjemalca za interakcijo z uporabnikom, nadzorom nad brskalnikom, asinhrono komunikacijo in spreminjanje prikazanje vsebine dokumenta. Uporablja se tudi za strežniško programiranje (Node.js), razvoj računalniških iger in izdelavo namiznih ter mobilnih aplikacij. [10]

jQuery

jQuery je brezplačna, odprtokodna in najpogosteje uporabljena knjižnica JavaScript, licencirana pod licenco MIT. Zasnovana je za poenostavitev ukaznih datotek odjemalca. Glavni prednosti knjižnice jQuery sta, da loči JavaScript in HTML in je zelo razširljiva, kar omogoča preprosto dodajanje novih dogodkov, elementov in metod. Primer razširitve jQuery v obliki vtičnika je graceful-websocket, ki poskrbi za implementacijo API spletne vtičnice in je kompatibilna tudi z brskalniki, ki ne podpirajo te tehnologije. [11]

Socket.io

Socket.io je knjižnica JavaScript na strani odjemalca, ki običajno deluje najbolje v povezavi z Node.js. Primerna je za razvoj spletnih aplikacij v realnem

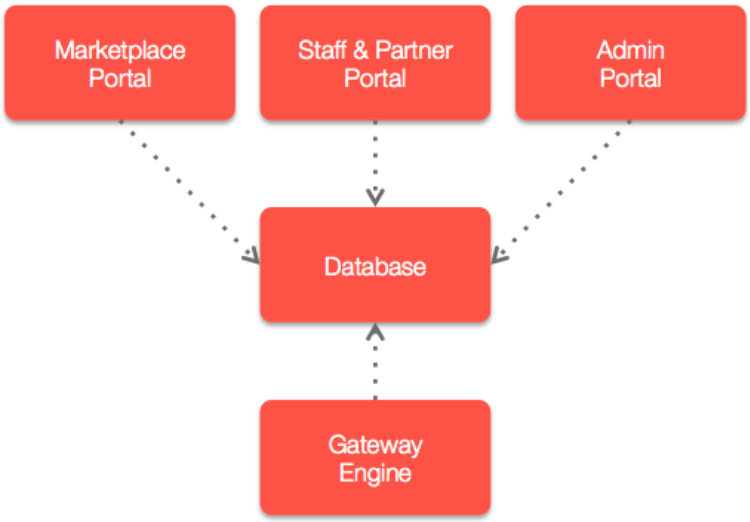
času. Omogoča obojestransko komunikacijo med spletnim strežnikom in odjemalcem. Socket.io je sestavljen iz dveh delov: knjižnice odjemalca, ki teče v brskalniku in knjižnice strežnika za Node.js. Obe komponenti imata skoraj identičen API. Socket.io primarno uporablja protokol WebSocket, ampak lahko po potrebi preide na uporabo starejše tehnologije (npr. AJAX long polling). [12]

Poglavje 3

Sistem za naročanje

Sistem za naročanje, ki predstavlja osnovo za nadgradnjo z uporabo potisne tehnologije, je spletna tržnica za podjetja, ki uporablja računalništvo v oblaku - internetno omrežje in gostovane strežnike za varno hranjenje uporabnikovih podatkov in izvajanje vseh aplikacij. Računalništvo v oblaku *(angl. cloud computing)* nudi prilagodljivo alternativo nakupu programske opreme in infrastrukture. Namesto nameščanja programov na lokalne računalnike in kupovanja strojne opreme lahko uporabniki uporabo programov le najamejo na strežnikih, ki sestavljajo računalniški oblak. Uporabniki tako do vseh rešitev dostopajo preko spletnega brskalnika. [13]

Sistem za naročanje je sestavljen iz petih komponent, ki omogočajo posredništvo storitev računalništva v oblaku. Pogled stranke na spletno tržnico predstavlja spletni portal, ki omogoča registracijo, naročilo storitev, pregled in urejanje aktivnih naročil ter pošiljanje zahtevkov za podporo. Portal za prodajalce in partnerje ter portal administratorjev je namenjen podpori in vzdrževanju interakcij podjetja s trenutnimi in bodočimi strankami. V ozadju se izvajajo storitve za razporejanje in izvajanje poteka dela in procesov. Najpomembnejša komponenta tega sistema pa je podatkovna baza, ki povezuje vse naštete komponente. Topologija programske rešitve je prikazana na Sliki 3.1. [14]



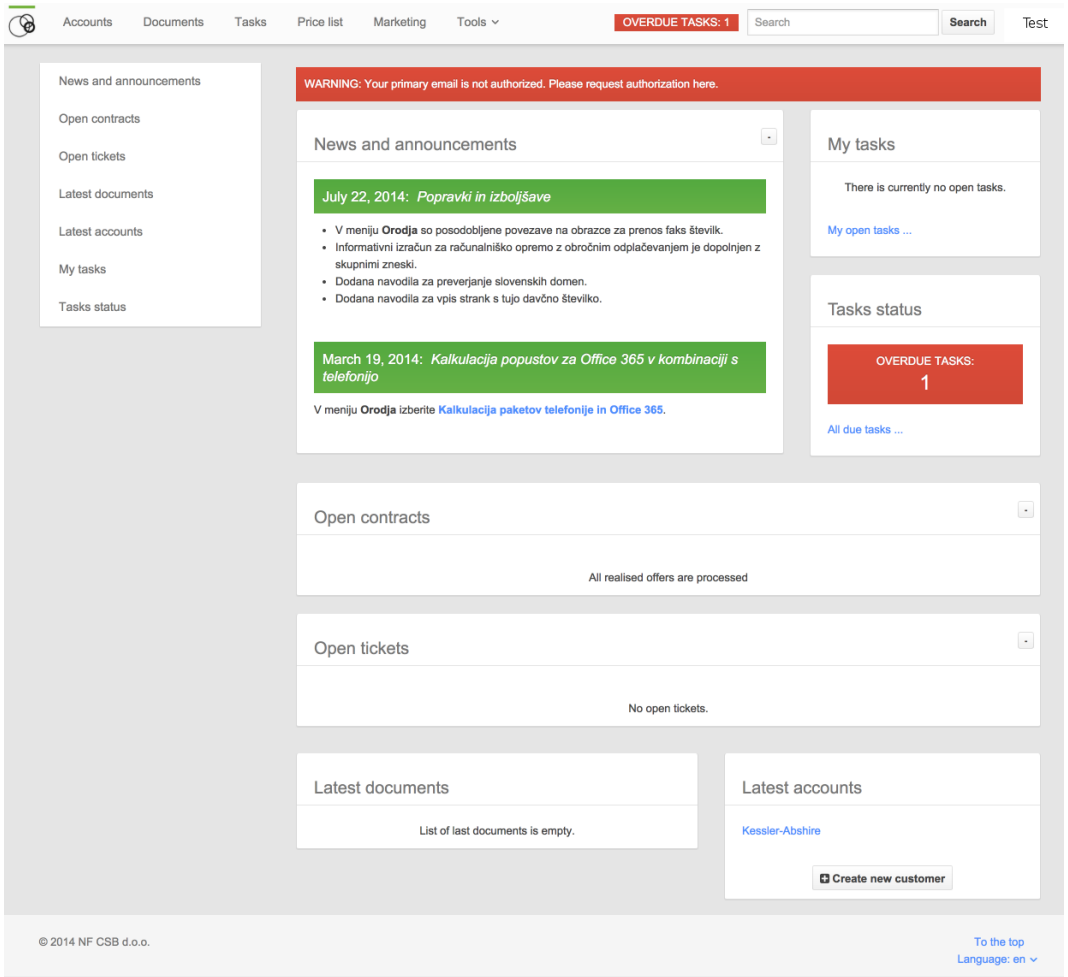
Slika 3.1: Topologija programske rešitve v produkcijskem okolju

Uporabniški vmesnik je prostor, kjer pride do interakcije med uporabnikom in računalnikom. V uporabniškem vmesniku bo v prehodu na produkcijski sistem tudi vidna implementacija potisne tehnologije. Uporabniški vmesnik sistema za naročanje se razlikuje glede na vrsto uporabnika in je prilagojen glede na ponujene funkcionalnosti - skrbniki sistema imajo na voljo več informacij kot registrirane stranke. Primer uporabniškega vmesnika je prikazan na Sliki 3.2.

Obstoječi sistem za naročanje je realiziran tako, da med oddajo naročila in izvedbo naročila lahko preteče dlje časa. Obstajata dve različici obdelovanja naročil; prva omogoča samodejno obdelavo naročila na strežniku, druga pa omogoča dodeljevanje naročil administratorjem, ki poskrbijo za določeno fazo naročila. Po vsaki obdelani fazi naročila se pošlje posodobitev statusa naročila po elektronski pošti.

Posodabljanje statusa naročil na spletni strani je sproženo na zahtevo preko odjemalca, zaradi česar uporabniška izkušnja ni najboljša. Prednost takšne rešitve je preprosta implementacija na strani odjemalca in strežnika, pomanjkljivost pa, da ne omogoča sprotnega in uporabniku prijaznega po-

sodabljanja statusa naročil. Slabost rešitve, ki samodejno pošilja poizvedbe na strežnik, je, da je njena natančnost odvisna od pogostosti poizvedovanja. Če želimo doseči vtis, da posodabljanje statusov poteka sproti, je potrebno prožiti večje število poizvedb, kar poveča obremenitev spletnega strežnika. Te poizvedbe v veliki večini tudi niso potrebne, ker od zadnje poizvedbe ni prišlo do spremembe stanja. Zato je izboljšana rešitev osnovana na temelju potisne tehnologije in omogoča sprotno posodabljanje stanja na strani odjemalca brez ustvarjanja nepotrebnega bremena na strani spletnega strežnika.



Slika 3.2: Zaslonski posnetek sistema za naročanje

3.1 Programska oprema

Sistem za naročanje teče na operacijskem sistemu Linux. Razvit je v programskem jeziku Python v ogrodju Django. Uporablja podatkovno bazo MySQL, poganja pa ga strežnik Apache. Uporabljena programska oprema je osnova za nadaljnji razvoj aplikacije, pri katerem je potrebno predvsem upoštevati kompatibilnost z obstoječo programsko opremo.

3.1.1 Django

Django je brezplačno odprtokodno ogrodje napisano v visokonivojskem programskem jeziku Python, namenjeno pisanju spletnih aplikacij. Začel se je začel razvijati leta 2003 in bil javno objavljen leta 2005 pod licenco BSD. Spletno ogrodje je bilo poimenovano po kitaristu Djangu Reinhardtu. Zasnovan je tako, da uporabniku olajša kreiranje kompleksnih spletnih strani v povezavi s podatkovno bazo. Poudarja ponovno uporabo komponent, hiter razvoj in princip DRY, ki stremi k zmanjševanju podvajanja programske kode.

Ogrodje Django temelji na arhitekturnem principu MVC. MVC je princip model-pogled-krmilnik za implementacijo uporabniških vmesnikov, ki razdeli programsko opremo na tri medsebojno povezane dele, tako da loči notranjo predstavitev informacije od načinov, kako je informacija predstavljena ali sprejeta od uporabnika. Jedro ogrodja Django MVC sestavlja:

- ORM, ki posreduje med podatkovnimi modeli (definiranimi kot razredi Python) in relacijsko podatkovno bazo (model),

- sistem za procesiranje zahtevkov s sistemom za predloge spletnih strani (pogled),

- odpremnik naslovov URL, ki temelji na regularnih izrazih (krmilnik). [15]

3.1.2 MySQL

MySQL je drugi najbolj pogosto uporabljeni odprtokodni sistem za upravljanje relacijskih podatkovnih baz. Izvorna koda je objavljena pod licenco GNU GPL. MySQL je osrednji sestavni del široko uporabljenega odprtokodnega programskega paketa za razvoj spletnih aplikacij, ki se imenuje LAMP in vključuje Linux, Apache, MySQL in Perl/PHP/Python. Podatkovna baza MySQL se uporablja za razvoj visoko zmogljivih in razširljivih aplikacij kot so Joomla, WordPress, Drupal in druge. MySQL uporabljajo tudi številne odmevne spletne strani velikega obsega - Google, Facebook, Twitter, Flickr in YouTube. [16]

3.1.3 Strežnik Apache

Strežnik Apache HTTP (httpd) je najpogosteje uporabljen odprtokodni spletni strežnik, ki zagotavlja varnost, učinkovitost in razširljivost ter storitve HTTP v sinhronizaciji s trenutnimi veljavnimi standardi HTTP. Apache se razvija pod okriljem Apache Software Foundation in je objavljen pod licenco Apache. Programska oprema je na voljo za številne operacijske sisteme. [17]

Poglavje 4

Implementacija

Pri izbiri tehnologij za implementacijo potisne tehnologije je predvsem potrebno upoštevati kompatibilnost in preprosto integracijo z obstoječim sistemom za naročanje. Poleg tega je potrebno imeti v mislih cilje, ki jih želimo doseči, število uporabnikov, ki bo uporabljalo sistem za naročanje in kasnejše možnosti razširitve sistema.

4.1 Razvojna orodja in tehnologije

Ker obstoječi sistem za naročanje uporablja ogrodje Django, je tudi pri izdelavi prototipa izbira ogrodja enaka. V razvojnem okolju sta uporabljena privzeta podatkovna baza SQLite in privzet razvojni strežnik Django, ki ju v produkcijskem okolju nadomestita podatkovna baza MySQL in strežnik Apache.

Pri izbiri spletnega ogrodja za realizacijo asinhronega delovanja spletne aplikacije na strani strežnika so imele prednost tehnologije, napisane v programskem jeziku Python, ker tako ni potreben prehod na nov programski jezik, kar omogoča lažje vzdrževanje kode. Zaradi dobre podpore, nadzora zmogljivosti, visoke razširljivosti, mnogih vgrajenih funkcionalnosti in številnih knjižnic smo se odločili za ogrodje Twisted, v katerem smo napisali spletni strežnik.

Na strani odjemalca smo za realizacijo potisne tehnologije poleg uporabe programskega jezika JavaScript vključili vtičnik jQuery *(angl. plugin)*, ki vrne objekt, ki implementira API spletne vtičnice. API spletne vtičnice je najsodobnejši vmesnik na področju potisne tehnologije, ki omogoča obojestransko komunikacijo TCP - lahko potisne podatke s strežnika na odjemalca kot tudi podatke z odjemalca na strežnika. Tako smo se izognili nepotrebnemu pošiljanju poizvedb na strežnik, ki je značilna za long-polling, kjer je učinkovitost delovanja odvisna od pogostosti poizvedovanja - če prožimo preveliko število poizvedb lahko preobremenimo strežnik. Pri takšnem načinu bi bilo veliko povezav odprtih in zaprtih po nepotrebnem, saj se od zadnje poizvedbe pogosto ne zgodi nikakršna sprememba stanja.

4.1.1 Podatkovna baza SQLite

Podatkovna baza SQLite je sistem za upravljanje relacijskih baz podatkov, vsebovan v programski knjižnici C. V nasprotju z ostalimi sistemi za upravljanje baz podatkov SQLite ni ločen proces, ampak sestavni del odjemalčeve aplikacije. SQLite je najširše uporabljena podatkovna baza, uporabljajo jo številni brskalniki, operacijski sistemi in vgrajeni sistemi. Izvorna koda za SQLite je v javni domeni.

Pri uporabi privzete podatkovne baze SQLite ogrodja Django je podatkovna baza shranjena v datoteki na računalniku. Ustvarjena je samodejno po potrebi in je ni potrebno konfigurirati. Za prikaz tabel v podatkovni bazi, ki jih je ustvaril Django, lahko zaženemo sqlite3 v ukazni vrstici in uporabimo ukaz *.schema*. [18]

4.1.2 Razvojni strežnik Django

Privzet razvojni spletni strežnik Django je napisan v programskem jeziku Python in teče na vratih 8000 na lokalnem naslovu IP 127.0.0.1. Ta strežnik ni primeren za produkcijsko okolje, ker še ni bil preizkušen na področju varnosti. Razvojni strežnik avtomatsko osveži kodo Python za vsak zahtevek,

tako da ni potrebno ponovno zaganjanje strežnika, da uveljavimo spremembo v kodi. Pri vsaki spremembi oziroma ob zagonu strežnika se validirajo vsi nameščeni modeli. V primeru, da strežnik najde napako, jo izpiše na standardni izhod, kar pa ne ustavi delovanja strežnika. Lahko zaženemo več strežnikov hkrati, potrebno je le paziti, da tečejo na različnih vratih. Za zagon strežnika izvedemo spodnji ukaz.

```
python manage.py runserver
```

Če želimo, da strežnik teče na izbranem naslovu IP in izbranih vratih in je tako dostopen tudi drugim napravam v omrežju, podamo naslov in vrata v argumentu. [19]

4.1.3 Strežnik Twisted

Strežnik Twisted nam omogoča realizacijo potisne tehnologije. Ogrodje Twisted uporabimo za implementacijo razčlenjevanja in ravnanja z omrežnim protokolom za strežnik TCP. Za zagon aplikacije Twisted uporabimo spodnji ukaz.

```
twistd -y imeaplikacije.py.
```

Ukaz twistd prebere vsebino *twisted.application.service.Application* iz datoteke in jo požene. Argument *-y* uporabi spremenljivko *application* iz dane datoteke Python *(imeaplikacije.py)*.

Razred, ki ravna s protokolom, se imenuje Protocol. Instanca razreda protokola se ustvari na zahtevo po povezavi in izgine, ko je povezava končana. To pomeni, da trajna povezava ni shranjena v razredu Protocol, ampak v razredu Factory. Metoda buildProtocol razreda Factory ustvari Protocol za vsako novo povezavo. Običajno je koristno, da lahko ponudimo enake storitve na več omrežnih naslovih ali vratih, zato Factory ne posluša povezav in ne ve ničesar o omrežju. Protokol Twisted obravnava podatke na asinhron način - protokol se odziva na dogodke iz omrežja, ki prispejo kot klici metod na protokolu. Strežnik posluša na vratih TCP tako, da požene dogodkovno zanko Twisted, ki čaka na povezave, ki prispejo na prej določena vrata.

Dogodkovna zanka se imenuje *reactor* in poganja aplikacije, ki uporabljajo Twisted. Reactor nudi vmesnike uporabniškega programa *(angl. API)* za omrežno povezovanje, nitenje *(angl. threading)*, razporejanje dogodkov in še mnoge druge. [20]

4.1.4 Vtičnik jQuery

Vtičnik jQuery *jquery-graceful-websocket* [21] implementira API spletne vtičnice. Namen tega vtičnika je, da omogoči razvijalcem uporabo vmesnika spletne vtičnice za pisanje aplikacij, tudi če spletni brskalnik tega ne podpira - vtičnik samodejno preide na uporabo starejše tehnologije (AJAX polling), ki uporablja HTTP POST za pošiljanje sporočil strežniku in zahtevke HTTP GET za prejemanje sporočil s strežnika. Z uporabo tega vtičnika se ni potrebno zanašati na lastniško tehnologijo, kot je recimo Flash, da preide na uporabo starejše tehnologije v primeru slabe podpore brskalnika.

4.1.5 JavaScript na strani odjemalca

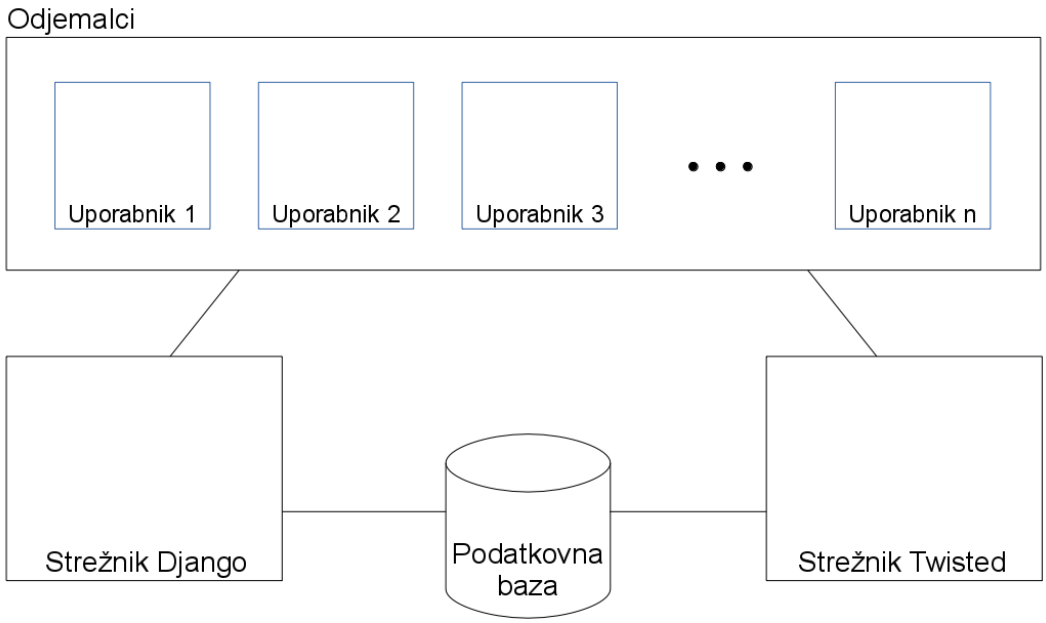
Koda JavaScript se uporablja za prikaz sporočil na strani odjemalca. Kodo se vključi v predlogo HTML uporabniškega vmesnika, kar omogoča asinhrono pošiljanje in prejemanje sporočil s strežnika. JavaScript uporablja vtičnik jQuery za ustvarjanje nove spletne vtičnice, na kateri lahko kliče funkcije za prejemanje in pošiljanje sporočil. Za pošiljanje sporočil in za določitev akcije ob prejemu sporočila uporabimo spodnja ukaza.

```
object.send(message)      #posiljanje sporocil
object.onmessage          #prejem sporocil
```

4.2 Prototip

Prototip je sistem, ki demonstrira delovanje potisne tehnologije v sistemu za naročanje. Sistem sestavljata dva strežnika. Strežnik Django je povezan s podatkovno bazo in poskrbi za prikaz osnovnega uporabniškega vmesnika.

Strežnik Twisted je prav tako povezan s podatkovno bazo in uporabniškim vmesnikom in poskrbi za realizacijo potisne tehnologije. Poleg tega s pomočjo ukazne lupine Python pišemo neposredno v podatkovno bazo, za kar bo v produkcijskem sistemu poskrbel strežnik. V zalednem delu sistema torej realiziramo večino potrebne logike za delovanje prototipa - branje in pisanje v podatkovno bazo ter potiskanje sporočil na odjemalce. Za prikaz sporočil v odjemalcu poskrbi JavaScript. Za namen demonstracije delovanja prototipa smo razvili preprosto podatkovno bazo in uporabniški vmesnik. Na Sliki 4.1 je prikazana skica sistema.



Slika 4.1: Skica sistema

Po zagonu obeh strežnikov (strežnika Django in strežnika Twisted) strežnik Twisted čaka na zapis v podatkovno bazo in v kratkih časovnih intervalih preverja, ali je prišlo do spremembe oziroma novega zapisa v podatkovni bazi. Za zapis v bazo bo v produkcijsem sistemu za naročanje poskrbel strežnik Apache, ki obdeluje naročila, v prototipu pa zapis v podatkovno bazo sprožimo kar v ukazni lupini. Ukazno lupino zaženemo s spodnjim

ukazom.

```
python manage.py shell
```

V podatkovno bazo zapišemo sporočilo, ki ga želimo poslati odjemalcem in dodamo uporabniška imena odjemalcev, ki jim to sporočilo želimo dostaviti. Z zapisom v podatkovno bazo sprožimo delovanje strežnika Twisted, ki zapisane podatke prebere iz baze in sporočilo potisne na odjemalce z navedenimi uporabniškimi imeni.

Nato strežnik Twisted v podatkovni bazi pobriše zapis v tabeli in tako ve, da je bilo sporočilo preneseno. Sporočilo je nemudoma prikazano v vseh odjemalcih z ujemajočim uporabniškim imenom. Poleg prejemanja sporočil lahko odjemalci tudi pošiljajo sporočila na strežnik in drugim uporabnikom, kar omogoča razvoj dodatnih funkcionalnosti, ki poskrbijo za večjo interaktivnost med uporabniki in boljšo uporabniško izkušnjo.

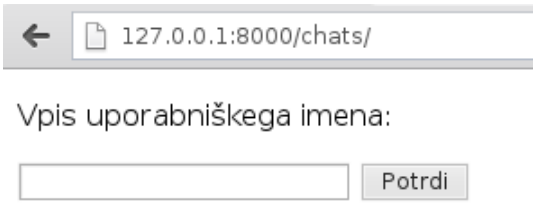
4.2.1 Uporabniški vmesnik

Uporabniški vmesnik prototipa je preprost, saj je namenjen izključno demonstraciji potisne tehnologije. V produkcijskem sistemu je uporabniški vmesnik že implementiran, tako da bo za namen implementacije potisne tehnologije v sistemu za naročanje potrebno samo določiti kam bodo prihajala sporočila. Poleg tega se lahko v obstoječi sistem implementira tudi možnost komunikacije v živo med uporabniki oziroma med uporabniki in administratorji. To funkcionalnost že nakazuje polje, kamor lahko uporabnik vpiše sporočila, ki jih želi poslati.

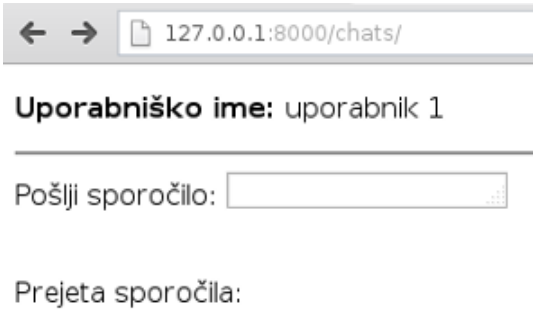
V prototipu se vsak uporabnik najprej prijavi z uporabniškim imenom, na podlagi katerega lahko strežnik filtrira sporočila in določi, katerim uporabnikom se bo določeno sporočilo prikazalo. Zaslonski posnetek prijave je prikazan na Sliki 4.2.

Po vpisu uporabniškega imena se prikaže vmesnik, ki prikazuje uporabniško ime. Ima vpisno polje, ki omogoča pošiljanje sporočil drugim uporabnikom in prostor, kamor prihajajo sporočila s strežnika. Uporabnik prejme

od strežnika le sporočila, ki so mu namenjena. Sporočila se samodejno prikazujejo drug po drugim brez osveževanja strani. Zaslonski posnetek uporabniškega vmesnika po prijavi je prikazan na Sliki 4.3.



Slika 4.2: Zaslonski posnetek: vpis uporabniškega imena



Slika 4.3: Zaslonski posnetek: uporabniški vmesnik

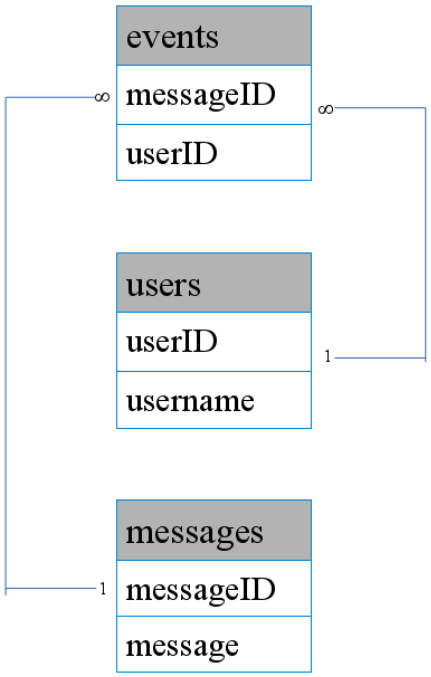
4.2.2 Podatkovna baza

Podatkovna baza je sestavljena iz treh tabel. V tabeli users imamo stolpca userID in username. V tabeli messages imamo stolpca messageID in message. Tabela events je povezovalna tabela in povezuje tabeli users in messages. V tej tabeli imamo stolpca userID in messageID.

V tabelo users shranjujemo uporabniška imena in ID (identiteto) uporabniškega imena. Uporabniška imena so predstavljena kot niz znakov, ID uporabniškega imena pa kot številka. V tabelo torej zapišemo uporabniška imena, ki jim želimo poslati sporočila.

V tabelo messages shranjujemo sporočila in ID sporočil. Sporočila so niz znakov, ID sporočil pa številka. V tej tabeli so shranjena sporočila, ki jih želimo poslati določenim uporabnikom.

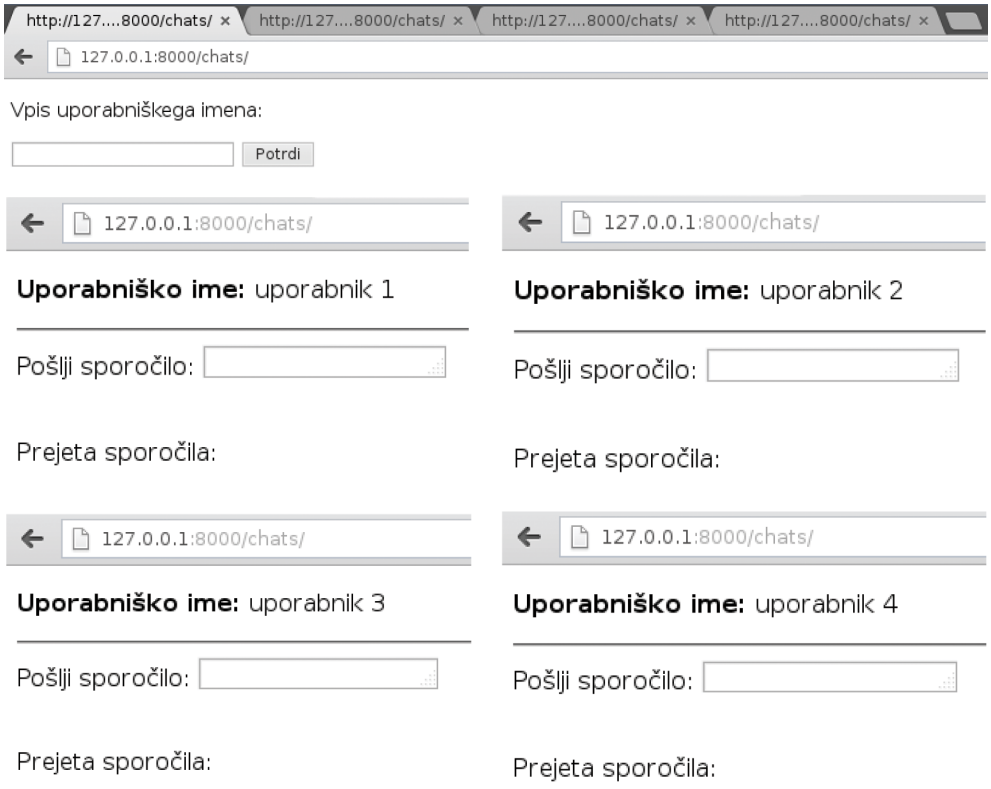
Tabela events povezuje tabeli users in messages v razmerju mnogo proti mnogo. Enemu uporabniku lahko pošljemo več sporočil, prav tako pa lahko eno sporočilo pošljemo več uporabnikom. Shematski prikaz celotne podatkovne baze je na Sliki 4.4.



Slika 4.4: Prikaz podatkovne baze

4.3 Primer delovanja prototipa

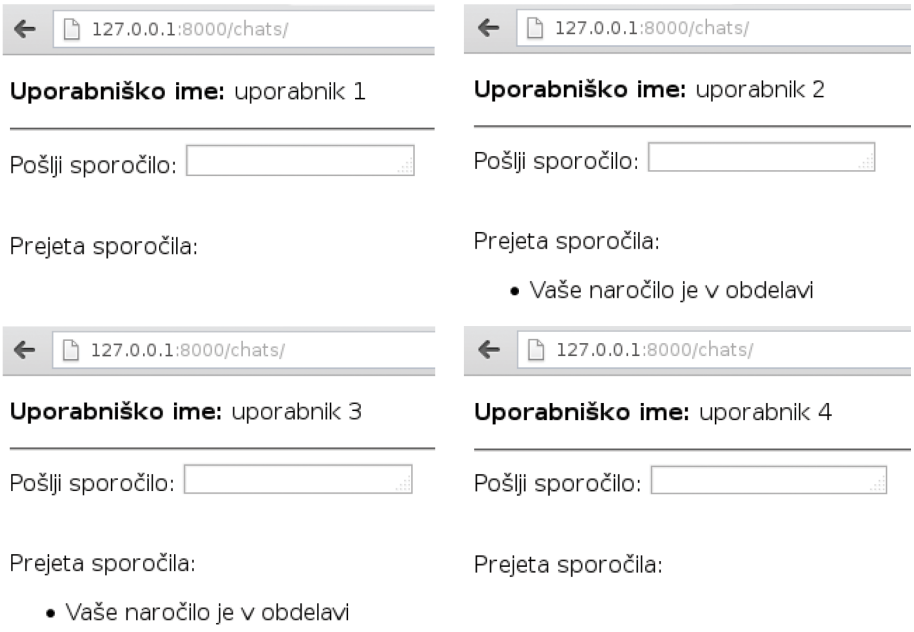
Za demonstracijo delovanja sistema najprej poženemo strežnika Django in Twisted. V brskalniku se povežemo na 127.0.0.1:8000 v štirih zavihkih. V prvem zavihku se prijavimo kot uporabnik 1, v drugem zavihku kot uporabnik 2, v tretjem kot uporabnik 3 in v četrtem kot uporabnik 4. Začetni prikaz vseh štirih uporabniških vmesnikov je na Sliki 4.5.



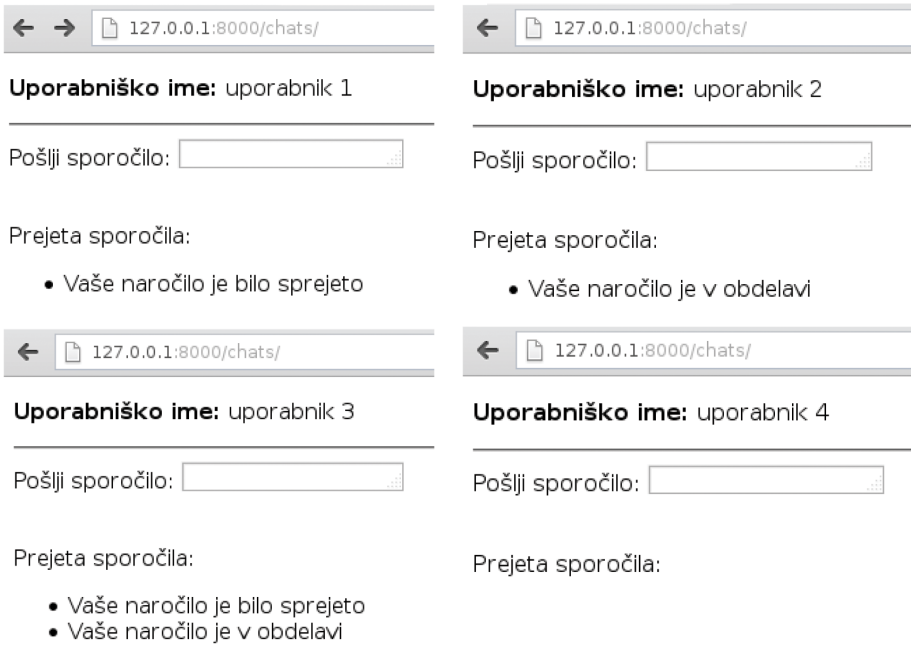
Slika 4.5: Začetni prikaz

Uporabnikoma 2 in 3 želimo poslati sporočilo 'Vaše naročilo je v obdelavi.' S pomočjo ukazne lupine zapišemo v podatkovno bazo uporabniški imeni obeh uporabnikov in sporočilo, ki jima ga želimo poslati. Strežnik Twisted prebere zapis v podatkovni bazi, sporočilo potisne na ustrezna odjemalca in pobriše zapis v bazi. Prikaz po prvem pošiljanju sporočila je na Sliki 4.6.

Nato želimo poslati uporabnikoma 1 in 3 sporočilo 'Vaše naročilo je bilo sprejeto.' Ponovno s pomočjo ukazne lupine zapišemo podatke v bazo, strežnik Twisted pa jih prebere in potisne na odjemalca. Uporabnik 3 je prejel dve sporočili, uporabnika 1 in 2 sta prejela po eno sporočilo, uporabnik 4 pa ni prejel nobenega sporočila. Prikaz po drugem pošiljanju sporočila je na Sliki 4.7.



Slika 4.6: Prikaz po pošiljanju prvega sporočila



Slika 4.7: Prikaz po pošiljanju drugega sporočila

Poglavje 5

Zaključki in nadaljnje delo

V diplomski nalogi je obravnavano področje spletnih storitev s potisno tehnologijo. Bil je narejen osnovni pregled programskih rešitev za realizacijo potisne tehnologije, na osnovi katerega je bila izbrana programska oprema za implementacijo rešitve. Praktični rezultat diplomske naloge je prototip, ki omogoča demonstracijo izboljšane uporabniške izkušnje s sprotnim posodabljanjem informacij o stanju naročil.

Ker je namen diplomske naloge nadgradnja obstoječega sistema za naročanje spletnih storitev s potisno tehnologijo, bomo izbrano programsko rešitev tudi vgradili v produkcijsko okolje. Povezali jo bomo z jedrom obstoječe rešitve, tako da bodo vse spremembe stanja naročil objavljene na novem komunikacijskem vodilu. Spletne strani, ki vključujejo informacijo o stanju naročil, bodo povezane s komunikacijskim vodilom in dopolnjene z ustrezno programsko logiko, ki bo skrbela za posodabljanje podatkov na osnovi sporočil prejetih prek komunikacijskega vodila.

Nadaljnje delo poleg implementacije v obstoječe okolje vključuje testiranje. Kljub temu, da je rešitev teoretično izredno razširljiva in kompatibilna z vsemi brskalniki, je potrebno to preveriti še v praksi. Rešitev mora zagotavljati normalno delovanje in hitro distribucijo podatkov tudi v okoliščinah, ko bo število aktivnih uporabnikov dosegalo okvire več tisoč, več deset tisoč ali več sto tisoč uporabnikov. Bistveno je, da je rešitev mogoče optimalno di-

menzionirati za sisteme z različno planirano obrementitvijo. Ker mora rešitev pred implementacijo zanesljivo delovati, je potrebno testirati tudi kompatibilnost spletne strani z različnimi spletnimi brskalniki.

Literatura

[1] Potisna tehnologija, dostopno na:
https://en.wikipedia.org/wiki/Push_technology

[2] Zgodovinski pregled potisne tehnologije, dostopno na:
https://cometdaily.com/2011/07/06/push-technology-comet

[3] Strežnik Comet, dostopno na:
https://www.stream-hub.com/about-comet.html

[4] Node.js, dostopno na:
https://nodejs.org/

[5] Twisted, dostopno na:
https://twistedmatrix.com

[6] Eventmachine, dostopno na:
https://en.wikipedia.org/wiki/EventMachine

[7] Tornado, dostopno na:
https://www.tornadoweb.org/

[8] Vert.x, dostopno na:
https://vertx.io/

[9] Libevent, dostopno na:
https://en.wikipedia.org/wiki/Libevent

[10] JavaScript, dostopno na:
https://en.wikipedia.org/wiki/JavaScript

[11] jQuery, dostopno na:
https://jquery.com/

[12] Socket.io, dostopno na:
https://en.wikipedia.org/wiki/Socket.IO

[13] Bizstore, dostopno na:
https://www.bizstore.si/o-bizstore/

[14] NFCSB. *WAGAmarket System Architecture Reference*. Revizija dokumenta 0.2. (08. 09. 2014)

[15] Django, dostopno na:
https://en.wikipedia.org/wiki/Django_(web_framework)

[16] MySQL, dostopno na:
https://www.mysql.com/

[17] Apache, dostopno na:
https://www.apache.org/

[18] SQLite, dostopno na:
https://en.wikipedia.org/wiki/SQLite

[19] Strežnik Django, dostopno na:
https://docs.djangoproject.com/en/1.3/ref/django-admin/

[20] Strežnik Twisted, dostopno na:
https://twistedmatrix.com/documents/12.1.0/core/howto/servers.html

[21] Vtičnik jQuery gracefulWebSocket, dostopno na:
https://code.google.com/p/jquery-graceful-websocket/